

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平 1 0 - 3 2 0 1 7 2

(43) 公開日 平成 1 0 年 (1 9 9 8) 1 2 月 4 日

(51) Int. Cl.

G06F 5/00

9/318

識別記号

庁内整理番号

F I

G06F 5/00

9/30

技術表示箇所

H

320

A

審査請求 未請求 請求項の数 6 O L (全 1 4 頁)

(21) 出願番号 特願平 9 - 1 2 5 9 8 6

(22) 出願日 平成 9 年 (1 9 9 7) 5 月 1 5 日

(71) 出願人 0 0 0 0 0 2 3 6 9

セイコーエプソン株式会社

東京都新宿区西新宿 2 丁目 4 番 1 号

(72) 発明者 小野 義之

長野県諏訪市大和 3 丁目 3 番 5 号 セイコーエプソン株式会社内

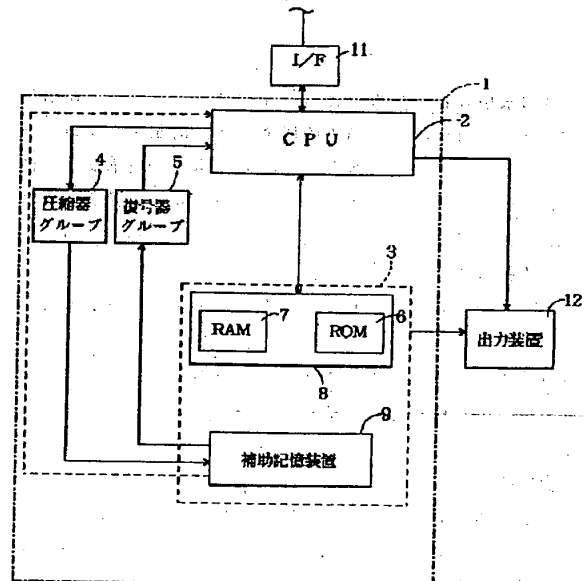
(74) 代理人 弁理士 鈴木 喜三郎 (外 2 名)

(54) 【発明の名称】 プログラム圧縮方法およびプログラム復号方法ならびにプログラム格納装置

(57) 【要約】

【課題】 中央処理装置を制御するためのマシンコード化されたプログラムの性質がどのようなものであろうと、一定の圧縮効果を得ること。

【解決手段】 プログラム格納装置 1 は、中央処理装置 2 と、中央処理装置 2 を動作させるためのマシンコードを圧縮する圧縮器グループ 4 と、圧縮器グループ 4 で圧縮されたマシンコード化されたプログラムを格納する記憶装置 3 と、記憶装置 3 内の圧縮されたプログラムを復号して中央処理装置 2 用のマシンコードとする復号器グループ 5 とを備え、命令の性質によって複数のタイプに区分されたマシンコードに対応して圧縮器グループ 4 のいずれか 1 つまたは複数を取捨選択してマシンコードを圧縮する。すなわち、中央処理装置 2 用のマシンコードを、それが意味する命令の性質によって複数のタイプに区分し、各タイプ毎に異なる圧縮アルゴリズムを使用してマシンコードを圧縮する。なお、復号時は逆のアルゴリズムを使用して復号する。



【特許請求の範囲】

【請求項1】 中央処理装置が制御する記憶装置に、マシンコード化されたプログラムを圧縮して格納するプログラム圧縮方法において、上記中央処理装置用のマシンコードを、それが意味する命令の性質によって複数のタイプに区分し、各タイプ毎に異なる圧縮アルゴリズムを使用して上記マシンコードを圧縮することを特徴とするプログラム圧縮方法。

【請求項2】 前記各命令タイプ毎に、そのマシンコードから少なくとも、レジスタフィールドおよび定数フィールドの2種類のデータを分離し、分離された各データをそれぞれ異なる符号化工程を使用して圧縮するようにしたことを特徴とする請求項1記載のプログラム圧縮方法。

【請求項3】 中央処理装置が制御する記憶装置から、圧縮されたマシンコードのプログラムを復号するプログラム復号方法において、圧縮されたマシンコードを、それが意味する命令の性質によって複数のタイプに区分し、各タイプ毎に異なる復号アルゴリズムを使用して、上記圧縮されたマシンコードを復号することを特徴とするプログラム復号方法。

【請求項4】 中央処理装置と、この中央処理装置を動作させるためのマシンコードを圧縮する圧縮器と、この圧縮器で圧縮されたマシンコード化されたプログラムを格納する記憶装置と、この記憶装置内の圧縮されたプログラムを復号して上記中央処理装置用のマシンコードとする復号器とを備えたプログラム格納装置において、上記圧縮器を複数設け、命令の性質によって複数のタイプに区分されたマシンコードに対応して上記複数の圧縮器のいずれか1つまたはいずれか複数を取捨選択してマシンコードを圧縮するようにしたことを特徴とするプログラム格納装置。

【請求項5】 少なくとも、前記マシンコードの命令フィールドを圧縮する第1の圧縮器と、前記マシンコードのレジスタフィールドを圧縮する第2の圧縮器と、前記マシンコードの定数フィールドを圧縮する第3の圧縮器とを有することを特徴とする請求項4記載のプログラム格納装置。

【請求項6】 中央処理装置と、この中央処理装置を動作させるためのマシンコードが圧縮されて格納された記憶装置と、この記憶装置内の圧縮されたプログラムを復号して上記中央処理装置用のマシンコードとする復号器とを備えたプログラム格納装置において、上記復号器を複数設け、命令の性質によって複数のタイプに区分されかつ圧縮されたマシンコードに対応して、上記複数の復号器のいずれか1つまたはいずれか複数を取捨選択して圧縮されたマシンコードを復号するようにしたことを特徴とするプログラム格納装置。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、マイクロプロセッサ等と呼ばれる中央処理装置を動作させるためのマシンコードからなるプログラムを圧縮する方法および圧縮されたプログラムを復号する方法ならびに圧縮されたプログラムを格納する装置の改良に関する。

【0002】

【従来の技術】中央処理装置は、その組み込まれた機器を制御するために、各種の制御や演算を行う。この制御等を行うに当たっては、付随して取り付けられる記憶装置等に格納されたプログラムに従っている。そして、このプログラムは、マシンコードとして作成されると共に、所定ビット構成に区切られて記憶装置内に格納されている。

【0003】通常、記憶装置がパラレルに入出力できるビット数は、4ビット、8ビット、16ビット等一定の長さ限定される。この長さは、1つのアドレスに格納できるデータのビット数に相当している。ところが、一般に、中央処理装置に対する命令は必ずしもすべて一定の長さのマシンコードで表現することができない。そこで、短いマシンコードは1つのアドレスに格納し、長いマシンコードは2つ以上のアドレスに振り分けて格納される。

【0004】したがって、コード長の長いマシンコードは、短いものに比べて読み出しに時間がかかり、このようなマシンコードが多いとプログラムの実行速度が低下してしまう。また、コード長の長い命令が多く含まれるプログラムは、これを格納するために大容量の記憶装置を必要とする難点もある。

【0005】これを解決するために、プログラムをマシンコード化して記憶装置に格納する場合において、プログラムに含まれる命令のうち、マシンコード化したときそのコード長が比較的長いものであって、そのプログラム中に比較的多く含まれている特定の命令について、そのマシンコードをこれより短い圧縮マシンコードに置換した後、記憶装置に格納する技術が知られている（特開昭61-201335号公報参照）。

【0006】また、所定回数連続するマシンコードを検出し、これを特定形式の圧縮コードに置き換えて記憶装置に格納するようにし、連続したマシンコードの格納領域を大幅に低減した装置も知られている（特開平3-110645号公報参照）。

【0007】他方、“0”と“1”からなる2値信号を圧縮する情報理論の世界では、算術符号化方式と呼ばれるものが知られている。この算術符号化方式は、エントロピー符号化方式であり、本質的に可逆符号化（ロスレス）の性質を持つものである。そして、その原理は、エライアスの符号化として知られている無記憶情報源に対する理想的符号化方式を実用可能な形に再編成したものとなっている。すなわち、算術符号とは、“0”と

50 “1”の直線上の対応区間を各シンボルの生起確率に応

じて不等長に分割していき、対象シンボル系列を対応する部分区間に割り当て、再帰的に分割を繰り返していくことにより得られた区間内に含まれる点の座標を、少なくとも他の区間と区別できる2進小数で表現してそのまま符号とするものである。

【0008】この算術符号化方式は、有限個の情報源シンボルに特定の符号語を対応させるブロック符号に比べ、符号器の規模が必要メモリ量などのハードウェアが小さくて済むこと、高い効率を期待できることおよび適応符号化が容易なこと等の利点がある。このこと等から、2値信号を扱う情報理論の世界では、この算術符号化方式がその情報の持つエントロピーに最も近いレベルに圧縮できるとされ、最も効率の良い符号化方式とされている。なお、この算術符号化方式は、特に、後述するマルコフ情報源の符号化に適するものとなっている。

【0009】この算術符号化方式として、Qコード、算術符号型MELコード、Mini-Maxコード等が提案されている。そして、これらの算術符号を改善したものとして、QMコードと呼ばれている方式が知られている。このQMコードは、カラー静止画符号化標準(JPEG)および2値画像符号化標準(JBIG)の両標準において、共通に使用されている。なお、このQMコードは、2値情報源用の符号であり、JPEGのような多値情報源の符号化にあたっては、その多値情報源を2値化するための前処理を必要としている。このような場合、符号化すべき2値シンボル数は増大するが、多値情報源としての情報量を増大させることなしに2値系列に変換することが可能となっている。

【0010】このQMコードは、JPEGおよびJBIGの規定の中にその仕組みについて詳細に述べられているが、ここでは後述する本発明を理解する参考として、その概要を図9に基づき簡単に説明する。なお、算術復号型のエントロピー復号器の構成は、エントロピー符号器の構成と実質的に同一であるので、ここではその説明は省略する。

【0011】この算術符号型のエントロピー符号器となるQMコード101は、算術演算部102と、状態記憶器として機能する発生確率生成手段103とを含んで構成される。この発生確率生成手段103内には、符号化に必要なシンボル発生確率を決定するために必要な状態パラメータテーブルが書き込まれている。上記の状態パラメータは、入力される状態信号106によって特定される。そして、この状態信号106によって特定された状態パラメータのテーブルに対し、発生確率手段103の発生確率演算パラメータが算術演算部102へ向け出力される。

【0012】算術演算部102は、このようにして入力される発生確率に基づき、エントロピー符号化を行い、入力されるデータ104を符号化データ105に圧縮し、符号化して出力する。そして、符号化されるデータ

104の値により、状態信号に対する発生確率を再計算し、演算パラメータ更新値が、発生確率生成手段103へ入力する。この更新結果が次データの発生確率としてテーブルに記憶されることで、QMコード101の圧縮効率が向上することとなる。なお、入力されるデータは2値化された信号であるが、その信号としては、文字情報や画像データや順位コード変換により変換された色順位データ(特開平6-276041号参照)等が代表例である。図10に、このような技術を多値情報源の代表例である画像データに用いた従来の符号化システム150および復号化システム160を示す。この符号化システム150は、ラインバッファ151と、エントロピー符号器152とを含むものである。入力されるインデックスの画素データ100Aは、ラインバッファ151およびエントロピー符号器152へ入力される。この画素データ100Aは、図11に示すように、いずれもラスタスキャンされ水平走査順に順次画素データとして入力される。

【0013】符号化システム150中のラインバッファ151は、参照画素生成手段として、既に入力された画素データ100Aから、符号化対象画素Xに対する参照画素データA、B、C、Dを作成する。すなわち、ラインバッファ151は、画像をスキャンするときにnライン(1~5ライン程度が多い)分の履歴を記憶しておく。そして、符号化対象画素Xの画素データ100Aが入力されるごとに、この直前の画素Aと、周辺の画素B、C、Dとからなる一連の画素データを参照画素データ110としてエントロピー符号器152へ向けて出力する。

【0014】このエントロピー符号器152は、QMコード101が使用されているが、ハフマン符号化などの手法を用いて形成したものとしても良い。そして、参照画素データ110(一種のコンテキスト)を状態信号として用い、対象画素データ100Aを符号化データ200に変換出力する。

【0015】一方、復号化システム160は、ラインバッファ161とエントロピー復号器162を含んで構成される。ここにおいて、ラインバッファ161とエントロピー復号器162は、入力される符号化データ200を符号化システム150のラインバッファ151、エントロピー符号器152とは全く逆の手順で復号化出力するように形成されている。

【0016】このようにして、符号化システム150と、復号化システム160とは、互いに全く逆のアルゴリズムを用いて、画素データ100Aを符号化データ200に符号化し、さらにこの符号化データ200を画素データ100Bに復号化して出力することができる。したがって、このシステムは、各種用途に幅広く用いられている。

【0017】しかし、このようなエントロピー符号器1

52およびエントロピー復号器162では、参照画素データの状態数に対応した数の符号化パラメータテーブルが必要となる。このため、圧縮率を高めるために参照画素数を大きくとればとるほど、符号化および復号化のパラメータテーブルが大きくなる。このため、エントロピー符号器152およびエントロピー復号器162が大型化かつ高価になってしまうという問題がある。

【0018】このような問題に対し、エントロピー符号器152およびエントロピー復号器162の中に縮退した状態数に応じてパラメータテーブルを小さくさせる技術が知られている。

【0019】この状態数を縮退するシステムの特徴は、図12に示すように、図10の符号化システム150や復号化システム160と同様にエントロピー符号器152およびエントロピー復号器162に参照画素データ110を状態信号として入力するわけであるが、その入力に際し、その状態信号140を、ラインバッファ151、161から出力される参照画素データ110を縮退する状態縮退器153、163によって生成する点にある。

【0020】この状態縮退器153、163は、入力される参照画素データ110を、より少ないビット数の状態信号140に縮退し、対応するエントロピー符号器152およびエントロピー復号器162へ向け出力するように構成されている。なお、予測器154、164は、処理する画素がカラーである場合、それぞれカラーシンボルの出現頻度に基づいてカラー画素データを色順位に変換するためおよびその逆を行うための色順位テーブルをそのメモリに保有しているものである。

【0021】なお、縮退とは、縮退後の状態数に、元の状態を分類する操作である。この分類は、分類後のエントロピー（1つのシンボルを表示するための平均情報量）が最少となるように、その組み合わせを選択して行う。そして、縮退後の状態数、すなわち、分類された後の状態数に対して識別ビットを付加する。これが状態信号140である。

【0022】なお、入力されてくるデータ、色順位データ120を生成するために、予測器154、164内に色順位テーブルが配置されている。この色順位テーブルの一例として、図13に示されるものが知られている

（特開平6-276041号公報参照）。この例では、符号化対象画素Xに対しての色順位テーブルを決める際、2次元の周辺画素データR0、R1、R2、R3を上位の色順位データとして使用し、符号化対象画素Xと同一ラインの1次元テーブルを下位の色順位データとして使用するものである。このとき、1次元テーブルから周辺画素データR0、R1、R2、R3のカラーシンボルを除去した後、上位と下位の色順位データをドッキングさせ符号化対象画素Xの色順位テーブルとしている。

【0023】具体的にどのように色順位テーブルができあがるかを図13に基づき説明する。符号化されるカラーシンボルが16色の場合を考える。仮に、色順位を図13(A)に示すように、各画素の位置R0、R1、R2…R8…で固定したとき、それぞれのカラーシンボルが図13(B)に示すように、C4、C3、C6、C5、C2、C2…のとき、できあがる最新出現表となる色順位テーブルは、図13(C)に示すようになる。すなわち、最上位はR0のC4、2番目はR1のC3、3番目はR2のC6、4番目はR3のC5、5番目はR4のC2、6番目はR5にあるC2となるが、C2は既に発生しており、さらにR6のC4も既に発生しているので、第6番目はR7のC0となる。このようにして既に上位にある色すなわち、R0～R3に出現するカラーシンボルを除いた色順位データがR0～R3のデータに加わり、16色のカラーシンボルの第1番目から16番目までが決められる。

【0024】

【発明が解決しようとする課題】このように、中央処理装置を制御するマシンコードを圧縮するには、特開昭61-201335号公報や特開平3-110645号公報に開示されるように、プログラムの特徴を検出して、圧縮する方法がある。また、その圧縮に利用する技術としては、上述したように、QMコード101等のエントロピー符号器、複数のレジスタからなるFIFOを利用した、いわゆる先頭移動処理(Move to Front)の最新出現表利用の技術、色順位変換等の順位コード変換技術、マルコフモデル等の状態縮退技術等が採用される。

【0025】しかしながら、本発明者の実験によれば、QMコード101や最新出現表や順位コード変換等の各方式のみでは、中央処理装置のマシンコードは十分圧縮されることが判明した。これは、その性質を無視して、単に8ビット等の単位で圧縮しているためである。この1バイト単位の圧縮は、LHA等のLZ系の圧縮の際にも行われている。

【0026】また、特開昭61-201335で開示される技術は、マシンコードのコード長の長い命令が多く含まれる特殊なプログラムには十分な効果があるものの、通常のプログラムに対しては圧縮効果はほとんど生じない。また、特開平3-110645の技術は、特定回数連続するコードを多く含むものに対しては、かなり効果があるものの、連続するコードがそれ程ないものについては、やはり圧縮の効果は出てこない。

【0027】本発明は、中央処理装置を制御するためのマシンコード化されたプログラムの性質がどのようなものであろうと、一定の圧縮効果を得ることができる新規なプログラム圧縮方法およびプログラム復号方法ならびにプログラム格納装置を提供することを目的とする。

【0028】

【課題を解決するための手段】かかる目的を達成するた

め、請求項1記載のプログラム圧縮方法では、中央処理装置が制御する記憶装置に、マシンコード化されたプログラムを圧縮して格納するプログラム圧縮方法において、中央処理装置用のマシンコードを、それが意味する命令の性質によって複数のタイプに区分し、各タイプ毎に異なる圧縮アルゴリズムを使用してマシンコードを圧縮している。

【0029】このように、中央処理装置を動作させる各コードが意味する命令の性質によって、複数のタイプにマシンコードを区分して、異なるアルゴリズムで圧縮するようにしているため、命令の性質に合った最適な圧縮が可能となる。このため、圧縮率が向上し、必要な記憶容量を低減させることができる。

【0030】また、請求項2記載の発明では、請求項1記載のプログラム圧縮方法において、各命令タイプ毎に、そのマシンコードから少なくとも、レジスタフィールドおよび定数フィールドの2種類のデータを分離し、分離された各データをそれぞれ異なる符号化工程を使用して圧縮している。

【0031】このため、マシンコードのタイプ毎の区分に加え、各フィールド毎に専用の圧縮器で符号化するので、一層、圧縮率が高いものとなる。そして、特に、ビット数が多くなるレジスタフィールドと定数フィールドをそれぞれ専用の符号化工程によって処理できるので他のフィールドの専用化に比べ圧縮率の面で有利となる。

【0032】また、請求項3記載のプログラム復号方法では、中央処理装置が制御する記憶装置から、圧縮されたマシンコードのプログラムを復号するプログラム復号方法において、圧縮されたマシンコードを、それが意味する命令の性質によって複数のタイプに区分し、各タイプ毎に異なる復号アルゴリズムを使用して、圧縮されたマシンコードを復号している。

【0033】このように復号に際し、その命令の性質に合った復号アルゴリズムを使用できるので、復号効率が向上する。また、圧縮されたマシンコードが保存される記憶装置の容量を低減させることができる。

【0034】さらに、請求項4記載の発明では、中央処理装置と、この中央処理装置を動作させるためのマシンコードを圧縮する圧縮器と、この圧縮器で圧縮されたマシンコード化されたプログラムを格納する記憶装置と、この記憶装置内の圧縮されたプログラムを復号して中央処理装置用のマシンコードとする復号器とを備えたプログラム格納装置において、圧縮器を複数設け、命令の性質によって複数のタイプに区分されたマシンコードに対応して複数の圧縮器のいずれか1つまたはいずれかの複数を取捨選択して、マシンコードを圧縮するようにしている。

【0035】このように、中央処理装置を動作させる各マシンコードが意味する命令の性質によって複数のタイプにマシンコードを区分して、各タイプ毎に対応する圧

縮器を使用しているので、命令の性質に合った最適な圧縮が可能となる。このため、この装置を使用すると、プログラムの性質に拘わらず、圧縮率が向上する。また、記憶容量も減少させることができ、この装置の高機能化も達成し易くなる。

【0036】加えて、請求項5記載の発明では、請求項4記載のプログラム格納装置において、少なくとも、マシンコードの命令フィールドを圧縮する第1の圧縮器と、マシンコードのレジスタフィールドを圧縮する第2の圧縮器と、マシンコードの定数フィールドを圧縮する第3の圧縮器とを設けている。

【0037】このため、マシンコードのタイプ毎の区分に加え、各マシンコードのフィールド毎に専用の圧縮器で符号化するので、一層、圧縮率が高いものとなる。

【0038】また、請求項6記載の発明では、中央処理装置と、この中央処理装置を動作させるためのマシンコードが圧縮されて格納された記憶装置と、この記憶装置内の圧縮されたプログラムを復号して中央処理装置用のマシンコードとする復号器とを備えたプログラム格納装置において、復号器を複数設け、命令の性質によって複数のタイプに区分されかつ圧縮されたマシンコードに対応して、複数の復号器のいずれか1つまたはいずれか複数を取捨選択して圧縮されたマシンコードを復号している。

【0039】このように、中央処理装置を動作させる圧縮された各マシンコードが意味する命令の性質によって複数のタイプにその圧縮されたマシンコードを区分して各タイプ毎に対応する復号器を使用しているため、命令の性質に合った最適な復号が可能となる。このため、この装置を使用すると、プログラムの性質に拘わらず復号効率が向上する。

【0040】本発明のプログラム圧縮方法およびプログラム復号方法ならびにプログラム格納装置では、中央処理装置が使用するマシンコードのくせを利用して、そのマシンコードを圧縮したり、復号したりしている。すなわち、中央処理装置の命令タイプによって、命令コード（一般にオペコードと呼ばれる）の後の部分の使われ方が違ってくるので、その命令タイプを判断して、命令タイプ毎に最適な圧縮方法や復号方法を採用している。よって、処理されるプログラムがどのような性質をもっていたとしても、一定の圧縮効果が発生し、復号効率も向上する。しかも、その圧縮率は、実験結果によると、LZ系の圧縮方式と同等レベルないしはそれ以上の圧縮率を得ることが出来るものとなっている。

【0041】

【発明の実施の形態】以下、本発明の実施の形態のプログラム格納装置1ならびにプログラム圧縮方法およびプログラム復号方法の第1の実施の形態について、図1から図6に基づき説明する。

【0042】このプログラム格納装置1は、中央処理装

置(以下、CPUという)2と、記憶装置3と、CPU 2を制御するプログラムを圧縮して記憶装置3に記憶させる複数の圧縮器からなる圧縮器グループ4と、記憶装置3に書き込まれた圧縮データを読み出して復号する複数の復号器からなる復号器グループ5とから主に構成される。ここで、記憶装置3は、リード・オンリ・メモリ(以下ROMという)6およびランダム・アクセス・メモリ(以下RAMという)7からなる主記憶装置8と、大量の情報を蓄える補助記憶装置9とからなる。

【0043】なお、このプログラム格納装置1には、CPU 2が、図示しないキーボードや外部記憶装置等と信号をやり取りするためのインターフェイス(I/F)11と、CRTや液晶パネル等で構成されCPU 2の制御演算結果を表示する出力装置12等が接続されている。また、圧縮器グループ4は、ハードではなく、いわゆるエンコードプログラムとして形成しても良く、復号器グループ5も同様に、デコードプログラムとして形成しても良い。

【0044】ここで使われているCPU 2の命令は、16ビット・フォーマットと32ビット・フォーマットの2種類となっている。そして、16ビット命令には、2項演算、制御、条件分岐などがあり、32ビット命令にはロード/ストア、I/O操作、16ビット・イミディエイトを扱う命令、ジャンプ・アンド・リンクなどがある。なお、一部の命令では、未使用フィールドが発生するが、それらは将来の拡張用で「0」に固定される。実際に命令がメモリに格納されるときは、次のように配置される。すなわち、各命令形式の下位部分(ビット0を含む)は、下位アドレス側に、各命令形式の上位部分(ビット15またはビット31を含む)は、上位アドレス側に配置される。

【0045】そして、各命令は、その性質によって図2に示すように、9種類のフォーマットでかつ計16種類のタイプに区分される。第1番目のAグループに属するものは、図2(A)に示すように、いわゆるreg-reg命令形式となっている。このタイプのものは、6ビットのオペコード・フィールド(=命令フィールド)とオペランド指定に2つの汎用レジスタ指定フィールド(=レジスタフィールド)をもつ命令形式となっており、16ビット長命令となっている。第2番目のBグループに属するものは、図2(B)に示すように、いわゆるimm-reg命令形式となっている。このタイプのものは、6ビットのオペコード・フィールドと、5ビットのイミディエイト・フィールド(=定数フィールド)と、1つの汎用レジスタ指定フィールドをもつ命令形式となっており、16ビット長命令となっている。なお、この実施の形態では、定数フィールドとは、命令フィールド、レジスタフィールドおよび後述する条件フィールドの計3つのフィールド以外のフィールドを指すものとしている。

【0046】第3番目のCグループに属するものは、図

2(C)に示すように、いわゆる条件分岐命令形式となっている。このタイプのものは、3ビットのオペコード・フィールド(=命令フィールド)と、4ビットの条件コードフィールド(=条件フィールド)と、9ビットの分岐ディスプレースメント・フィールド(=定数フィールド)と、1ビットのサブオペコードフィールド(=定数フィールド)をもつ命令形式となっており、16ビット長命令となっている。なお、9ビットの分岐ディスプレースメント・フィールドでは、「ビット0」は「0」とみなし、指定しないようになっている。また、第4番目のDグループに属するものは、図2(D)に示すように、いわゆる中距離ジャンプ命令形式となっている。このタイプのものは、6ビットのオペコード・フィールドと、26ビットのディスプレースメント・フィールドをもつ中距離分岐命令形式となっており、32ビット長命令となっている。

【0047】第5番目のEグループに属するものは、図2(E)に示すように、いわゆる3オペランド命令形式となっている。このタイプのものは、6ビットのオペコード・フィールドと、2つの汎用レジスタ指定フィールドと、16ビット・イミディエイト・フィールドをもつ命令形式となっており、32ビット長命令となっている。さらに、第6番目のFグループに属するものは、図2(F)に示すように、いわゆるロード/ストア命令形式となっている。このタイプのものは、6ビットのオペコード・フィールドと、2つの汎用レジスタ指定フィールドと、16ビット・ディスプレースメント・フィールドをもつ命令形式となっており、32ビット長命令となっている。

【0048】さらに、第7番目のGグループに属するものは、図2(G)に示すように、いわゆる拡張命令形式となっている。このタイプのものは、6ビットのオペコード・フィールドと、2つの汎用レジスタ指定フィールドと、6ビットのサブオペコード・フィールド(=定数フィールド)をもつ命令形式となっており、32ビット長命令となっている。また、第8番目のHグループに属するものは、図2(H)に示すように、いわゆる3レジスタ・オペランド命令形式となっている。このタイプのものは、6ビットのオペコード・フィールドと、3つの汎用レジスタ指定フィールドと、6ビットのサブオペコード・フィールドをもつ命令形式となっており、32ビット長命令となっている。なお、図2(G)中で示される「RFU」は、「Reserved for Future Use」の略で、本実施の形態では、未使用部分とみなして圧縮および記憶せずに捨ててしまっている。

【0049】第9番目のIグループに属するものは、図2(I)に示すように、いわゆるオペランドなし命令形式となっている。このタイプのものは、6ビットのオペコード・フィールドと、1ビットのサブオペコード・フィールドをもつ命令形式となっており、16ビット長命

10

30

40

50

令となっている。その後のJグループからPグループのものは、それぞれ前述の9種類のフォーマットのいずれかと同様であるが、その命令の性質によって違うグループに区分されるものである。

【0050】以上のように、CPU2の命令がその性質によって複数のタイプに区分される。そして、その各タイプによって使用される圧縮器が異なるものとなる。すなわち、圧縮器グループ4は、第1の圧縮器となる命令フィールド（オペコード・フィールド）用の圧縮器と、第2の圧縮器となるレジスタフィールド（汎用レジスタ指定フィールド）用の圧縮器と、第3の圧縮器となる条件フィールド（条件コード・フィールド）用の圧縮器と、第4の圧縮器となる定数フィールド用の圧縮器の計4つの圧縮器から構成され、命令タイプによって使用する圧縮器を図3に示すように異ならせている。

【0051】なお、各圧縮器としては、ここでは、算術符号式のエントロピー符号器、具体的には図9で示したようなQMコードを使用しているが、他の圧縮器を使用しても良い。同様に各復号器は算術復号型のエントロピー復号器を使用しているが、他の復号器を使用しても良い。なお、第1の圧縮器および第1の復号器は、命令フィールドが6ビットとなっているが、命令タイプとしては16種類に区分されているため、4ビットで全タイプを現すことができ、4ビット幅を有するQMコードとしている。また、第2の圧縮器および第2の復号器は、レジスタフィールドが5ビットとなっているので、5ビット幅を有するQMコードとしている。同様に条件フィールド用の第3の圧縮器および第3の復号器は、4ビット幅のQMコードで、定数フィールド用の第4の圧縮器および第4の復号器は、1ビット幅のQMコードとしている。

【0052】次に、このように構成されるプログラム格納装置1の圧縮格納動作について、図4等に基づき説明する。

【0053】まずは、CPU2がインターフェイス11を介してまたは補助記憶装置9内に取り込まれていた格納すべきプログラムを読み出す。インターフェイス11を介する場合の例としては、通信等により特定のプログラムをダウンロードする場合やキーボードによって直接プログラムを生成する場合等がある。そして、そのプログラムを記憶装置3の一部を構成する補助記憶装置9に保存するとき、この圧縮処理がスタート（ステップS1）する。

【0054】最初、プログラムに書かれたCPU2の命令タイプがAグループに属するものか否かを判断する（ステップS2）。そして、Aグループに属するものであるときは、Aグループの圧縮器、具体的には図3に示すように、命令フィールド用の第1の圧縮器と、レジスタフィールド用の第2の圧縮器とを使用して圧縮する（ステップS3）する。

【0055】この圧縮の際、過去のデータの情報を各QMコードにコンテキスト（＝状態信号）として入力する。これによって圧縮率をさらに高めている。なお、このコンテキスト入力は、各圧縮の際、すべてのQMコードに入力するようにしている。ただし、後述するように、コンテキストを入力しない場合でも相当な圧縮率を得ることができる。

【0056】そして、圧縮されたそのマシンコードを記憶装置3、この実施の形態では、補助記憶装置9に書き込む（ステップS4）。その後、そのプログラムが終了か否かを判断して（ステップS5）、終了の場合はストップする（ステップS6）。終了でないときは、次のマシンコードを処理するためステップS2に戻る。

【0057】一方、最初の命令タイプがAグループに属さないときは、Bグループに属するか否かを判断して（ステップS7）、Bグループに属するときは、Bグループの圧縮器によってそのマシンコードを圧縮する（ステップS8）。具体的には、図3に示すように、命令フィールド用の第1の圧縮器と、レジスタフィールド用の第2の圧縮器と、定数フィールド用の第4の圧縮器とを使用して圧縮する。そして、圧縮されたマシンコードを同様に記憶装置3に書き込む（ステップS9）。この後、プログラムが終了か否かを判断して（ステップS5）、終了であればストップし（ステップS6）、終了しない場合は、次のマシンコードを処理するためステップS2に戻る。

【0058】最初の命令タイプがAグループにもBグループにも属さない場合、Cグループに属するか否かを判断して（ステップS10）、Cグループに属するときは、Cグループの圧縮器によってそのマシンコードを圧縮する（ステップS11）。具体的には、図3に示すように、命令フィールド用の第1の圧縮器と、条件フィールド用の第3の圧縮器と、定数フィールド用の第4の圧縮器とを使用して圧縮する。そして、圧縮されたマシンコードを同様に記憶装置3内の補助記憶装置9に書き込む（ステップS12）。以下、上述したステップと同様なステップを繰り返して圧縮し格納していく。

【0059】なお、この圧縮においては、特に、レジスタフィールドに書かれるレジスタ番号や定数フィールドに書かれる分岐命令の圧縮の際に効果が大となる。これは、レジスタ番号や分岐命令部分がビットを多く必要とする部分であるためである。特に、レジスタ番号は、同じものがたびたび現れることが多いため、一度現れてきたものを複数個、具体的にはこの実施の形態では2個残しておき、コンテキストとして入力させることにより圧縮率が高まることとなる。なお、2個ではなく、3個以上としても良く、また、このコンテキストとしての入力の際、図13に示すような最新出現表を作成し、利用すると一層、圧縮率が向上する。

【0060】また、分岐命令の場合、現在位置から現し

た相対位置、例えば「現在位置から20番地前の位置へ移れ」等の相対位置で表現されることが多い。このような場合、プログラムが進んで、現在位置がどんどん進んでいくと、戻る番地は同じであっても表現は異なるものとなる。例えば、先の例で言えば、プログラムが3番地進んでいると、「現在位置から23番地前の位置へ移れ」という表現となる。これは数字が変わることになり、圧縮にとって不利な表現である。よって、この実施の形態では行っていないが、プログラムが進むごとに現在位置を覚えておいて、すなわちプログラムが進むごとに現在地を補正して、元のアドレスを常に圧縮するようにするのが好ましい。

【0061】なお、この分岐命令の場合においても、過去の履歴を覚えておいて、その情報をコンテキストとして入力するようにしても良い。この際、相対番地として覚えておくか、絶対番地として覚えておくかは、適宜選択することができる。また、過去の履歴としては、先程のように先頭移動処理を行うようにしても良い。

【0062】また、オペコード(opcode)の部分は6ビットで表現されるが、6ビットすなわち $2^6 = 64$ 種類の命令があるかというそうではなく、64種類未満の命令となっている。通常の圧縮であると、このあり得ない命令も通常の処理によって圧縮するが、この実施の形態では命令タイプの性質によって区分しているので、あり得ない命令が出てくると、ストップさせるか通常の圧縮を行うようにできる。

【0063】さらに、未定義命令が発生するような場合、その命令を認識すると、その後のビットは切り捨ててしまうようなこともできる。さらには命令タイプによっては、その後に生ずるビットが確定しているものを生ずることがある。このような場合、例えば、テーブルを持っておいて、対応するその旨の符号のみを記憶させておき、復号するときに、そのテーブルから該当する確定ビットを持つてくるようにすることもできる。

【0064】この実施の形態のプログラム格納装置1における圧縮率は、図5に示すとおりとなっている。すなわち、コンテキストを利用せず、単に分類したときでも約75.0%の圧縮率を示している。その値は、LZ系の基本アルゴリズムを使用したときの値よりは6%程度悪くなっているが、それでもかなりの圧縮率となっている。

【0065】一方、コンテキストを図6に示すような形で利用したときは69.6%となり、LZ系とほとんど同じとなっている。このコンテキスト利用は、単純に直前の過去のデータの情報を利用するだけのもので、しかも図6に示すように、レジスタフィールドと定数フィールドのみしか利用していない。このような利用程度でもかなりの圧縮率となっている。よって、QMコードにマシンコードを入れる前に、先に示した先頭移動処理や相対位置の記憶等、各種の改善策によって処理すれば、さ

らに一層良い圧縮率が得られることが想定される。

【0066】なお、命令フィールドは、今回、その性質に併せて16種類に人手によって分けたが、統計をとってかつ自動的に分けるようにすることが好ましい。また、レジスタフィールドは、先頭移動処理を行ってからQMコードに入れるようにするのが好ましい。さらに、条件フィールドは、オペコードの条件とからめて予測するのが好ましい。定数フィールドのアドレスに対しては、相対番地を絶対番地に変えて、それを記憶させる一種の先頭移動処理をやらせるようにしたり、ハフマンコードのような処理の仕方を行うようにしても良い。

【0067】また、プログラム中に生のデータがある場合は、そのプログラム中にフラグを立て、そのフラグを検知して汎用のアルゴリズムに切り替えるようにするのが好ましい。この場合、そのデータ部分については、先頭移動処理や単なるQMコードやLZ系の圧縮方法を採用するようにする。また、コンパイラによってデータ部分を分離するようにしても良い。

【0068】なお、ネットワーク中でプログラムを移動させたり、アップロードや、ダウンロードするときに、課金がなされるネットワーク部分は圧縮して、小さい容量にして移動させ、その他の部分は復号して移動させるようにしても良い。例えば、プロキシサーバにこの圧縮復号機能を持たせ、自動的にフィルタリングするようにする技術を採用できる。このようにすることにより、大容量のプログラムを短時間で送信したり、アップロードやダウンロードを短時間かつ低費用で行えるようになる。しかも、作業者は、圧縮する際、特別な作業を必要としないので、煩わしさがなくなる。

【0069】なお、このCPU2の命令タイプは、図2のように、16種類に区分できるが、マシンコードのフォーマットや性質は、各CPUやコンパイラによって全く異なるため、CPUが異なればその区分けの仕方も全く異なるものとなる。すなわち、各CPU毎に異なる処理工程および異なる圧縮器や復号器が必要となる。

【0070】次に、本発明の第2の実施の形態のプログラム格納装置2を図7に基づいて説明する。このプログラム格納装置2で使用している圧縮方法や好ましい圧縮方法については、第1の実施の形態のプログラム格納装置1と同様である。また、説明に当たり、第1の実施の形態のプログラム格納装置と同一部材には同一符号を付して、その説明を省略または簡略化することとする。

【0071】この第2の実施の形態のプログラム格納装置2では、圧縮器グループ4と復号器グループ5をCPU2と主記憶装置8との間に配置している。このため、ROM6に書き込まれる、CPU2と密接不可分なプログラムが圧縮して保存されたり、インターフェイス11等から入力され一旦RAM7に書かれるマシンコードも圧縮して保存されることとなる。このため、ROM

6やRAM 7の容量を小さなものにでき、同じ容量のROM 6やRAM 7を採用すると、大容量のプログラムの処理が可能となる。

【0072】この第2の実施の形態でも、圧縮器グループ4や復号器グループ5をそれぞれハードではなく、エンコードプログラムやデコードプログラムに置き換えても良い。この場合、各プログラムを、圧縮されたプログラムが格納されるROM 6とは別のROMに書き込んだり、ROM 6を分割して、分割された部分に書き込んでおくのが好ましい。そして、その別のROM内のエンコードプログラムやデコードプログラムを使用して圧縮したり復号したりするようにする。また、ROM 6内に書き込まれる当初のプログラムは圧縮せず通常の形態とし、RAM 7に保存されるマシンコードのみを圧縮するようにしても良い。

【0073】次に、本発明の第3の実施の形態のプログラム格納装置41を図8に基づいて説明する。このプログラム格納装置41で使用している圧縮方法や好ましい圧縮方法についても第1の実施の形態のプログラム格納装置1と同様である。また、説明に当たっては、第1の実施の形態のプログラム格納装置と同一部材には同一符号を付して、その説明を省略または簡略化することとする。

【0074】この第3の実施の形態のプログラム格納装置41では、CPU 2と主記憶装置8との間に復号器グループ5のみを配置している。ROM 6へのプログラムの圧縮書き込みは、CPU 2と同様な動作をするプログラムおよびエンコードプログラムによって図示しない別ルートで行う。このため、ROM 6には、この発明の方法で圧縮されたプログラムが既に格納されている。CPU 2は、復号器グループ5によって復号されたプログラムによって動作するようになる。このようなプログラム格納装置41は、特定の用途に特化されたCPU 2、例えば電気炊飯器や電気冷蔵庫等の家庭用電気製品や、ロボット等の工業製品等に使用されるCPU-2に適用して好適となる。

【0075】なお、上述の各実施の形態は、本発明の好適な実施の形態の例であるが、これに限定されるものではなく、本発明の要旨を逸脱しない範囲において、種々変形実施可能である。例えば、第1の実施の形態のプログラム格納装置1と、第2の実施の形態のプログラム格納装置21とを合わせた構成、すなわち圧縮器グループ4と復号器グループ5を、CPU 2と補助記憶装置9との間およびCPU 2と主記憶装置8との間の両部分に配置するようにしても良い。

【0076】また、本発明は、命令の分類分けが16種類のようにある程度少ない数に分けることができるものに適用して好適であるが、30種類や60種類のように多くの種類に分類分けされるものにも適用できる。

【0077】また、上述の実施の形態では、4つのQM

コードを使用した、4つ以外の数としたり、少なくとも、ビット数が多くなりがちなレジスタフィールド用および定数フィールド用の2つの圧縮器や復号器を使用することにより、プログラムの圧縮率を相当高めることができる。加えて、QMコード以外の各種の圧縮や復号のアルゴリズムを採用することができる。

【0078】さらには、圧縮器や復号器を1つとし、命令タイプの性質によって区分した表をコンテキストとして利用するだけでも圧縮率は向上する。また、条件フィールドを圧縮する際、命令フィールドの性質とからませて一種の状態縮退器(図12参照)を形成してその状態をコンテキストとしてQMコード等の圧縮器に入力するようにしても良い。さらには、分離されたフィールドの性質によっては、圧縮せず、そのまま記憶させるようにしても良い。また、各圧縮器や各復号器に、特定データについては、圧縮や復号をせず、入力されてきたデータをそのまま素通しさせる、いわゆる素通し機能を持たせるようにしても良い。

【0079】

【発明の効果】以上説明したように、請求項1および2記載のプログラム圧縮方法では、CPUの命令の性質によって、その後に続くマシンコードが所定の配置となる、いわゆるCPUのくせを利用して圧縮している。このため、プログラムの内容に拘わらずそのプログラムを高圧縮率なものとすることができる。

【0080】また、請求項3記載のプログラム復号方法では、CPUの命令の性質によって、その後に続く圧縮されたマシンコードが所定の配置となる、いわゆるCPUのくせを利用して復号している。このため、圧縮されたプログラムの内容に拘わらずそのCPUにとって最適かつ高効率にて復号することができる。

【0081】さらに、請求項4から6記載のプログラム格納装置では、CPUの命令の性質によって、その後に続くマシンコードが所定の配置となる、いわゆるCPUのくせを利用して圧縮したり復号したりしている。このため、プログラムをそのCPUにとって最適かつ高効率で圧縮したり復号したりできる。加えて、記憶装置の容量が減少し、低価格化が可能となると共に、同じ容量の記憶装置を使用すると、大容量のプログラムを保存することが可能となる。

【図面の簡単な説明】

【図1】本発明の第1の実施の形態のプログラム格納装置のデータの流れを示すブロック図である。

【図2】図1のプログラム格納装置に使用されるCPUの命令タイプを示す図で、(A)から(P)の計16種類のタイプに区分されることを示す図である。

【図3】図1のプログラム格納装置に使用されるCPUの命令タイプと、その命令タイプのときに使用する圧縮器との関係を示す図である。

【図4】図1のプログラム格納装置で使用されるプログ

ラム圧縮方法を説明するためのフローチャートである。

【図5】図1のプログラム格納装置および図4のフローチャートによって対象プログラムが圧縮された結果を示す図で、参考に従来の方法による圧縮率も併せて示した図である。

【図6】図1のプログラム格納装置および図4のフローチャートによって対象プログラムを圧縮する際のコンテキストの利用状況を説明するための図である。

【図7】本発明の第2の実施の形態のプログラム格納装置のデータの流れを示すブロック図である。

【図8】本発明の第3の実施の形態のプログラム格納装置のデータの流れを示すブロック図である。

【図9】従来の算術符号型のエントロピー符号器およびエントロピー復号器の説明図である。

【図10】従来の符号化システムおよび復号化システムのブロック図である。

【図11】従来の状態信号の生成を説明するための図で、符号化対象画素データに対する参照画素データの説

明図である。

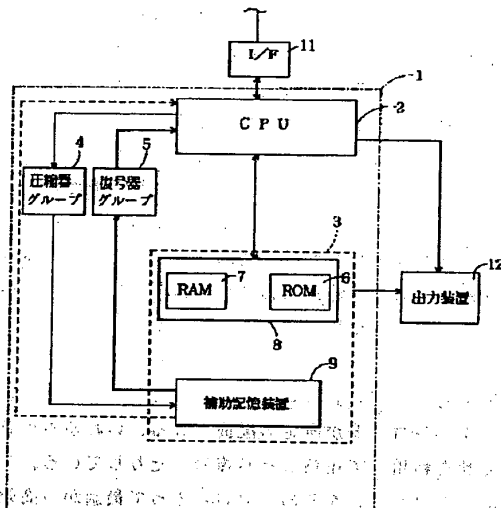
【図12】状態縮退器を有する従来の符号化システムおよび復号化システムのブロック図である。

【図13】従来の最新出現表の使用例を説明するための図で、(A)は入力されてくる画素と符号化対象画素の位置関係を示し、(B)は、各画素のカラーシンボルを示し、(C)は最新出現表を示す図である。

【符号の説明】

- 1 プログラム格納装置
- 10 2 CPU (中央処理装置)
- 3 記憶装置
- 4 圧縮器グループ
- 5 復号器グループ
- 6 ROM (リード・オンリ・メモリ)
- 7 RAM (ランダム・アクセス・メモリ)
- 8 主記憶装置
- 9 補助記憶装置

【図1】



【図3】

○: 使用を示す

命令 タイプ	使用する圧縮器			
	命令フィールド用	レジスタフィールド用	条件フィールド用	定数フィールド用
Aグループ	○	○		
Bグループ	○	○		○
Cグループ	○		○	○
Dグループ	○			○
Eグループ	○	○		○
Fグループ
Iグループ	○	○	○	○
...

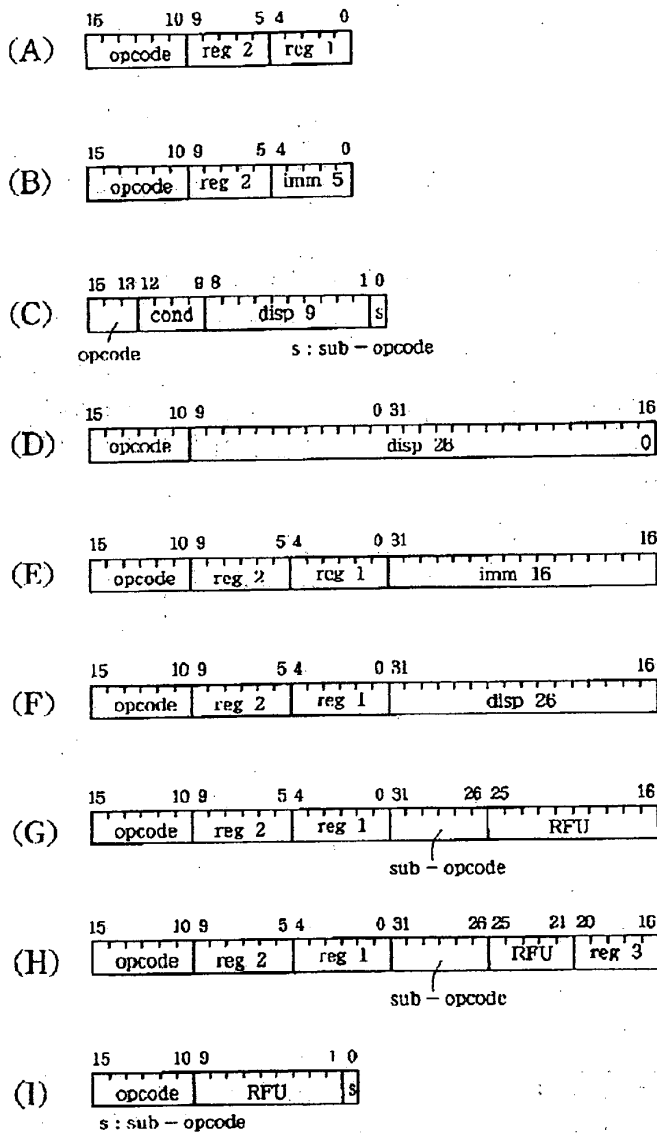
【図5】

方式		対象プログラム (2,000バイト)
		圧縮率
参考	LZ系基本アルゴリズム	69.0 %
本発明	4+5+4+1ビットQMコード	75.0 %
	コンテキスト採用 4+5+4+1ビットQMコード	69.6 %

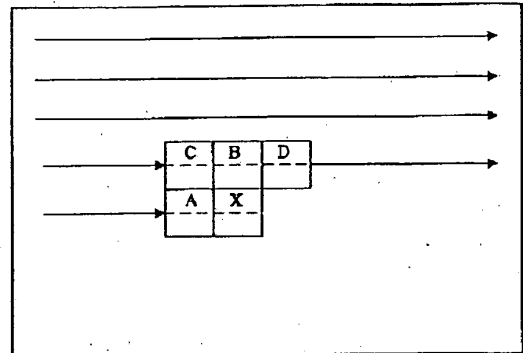
【図6】

コンテキスト (過去の単純データ)		
命令フィールド	4ビット×0個	※コンテキスト利用しない
レジスタフィールド	5ビット×2個	直前2個分の確率統計を取る
条件フィールド	4ビット×0個	※コンテキスト利用しない
定数フィールド	1ビット×8個	単純に過去8個分を

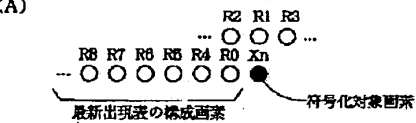
【図 2】



【図 11】



【図 13】



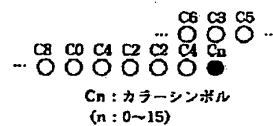
画素優先順位

学習なし: R0, R1, R2...R8... (全画素固定)

学習機能: R0, R1, R2, R3, R4, R5...R8...

学習により可変

固定

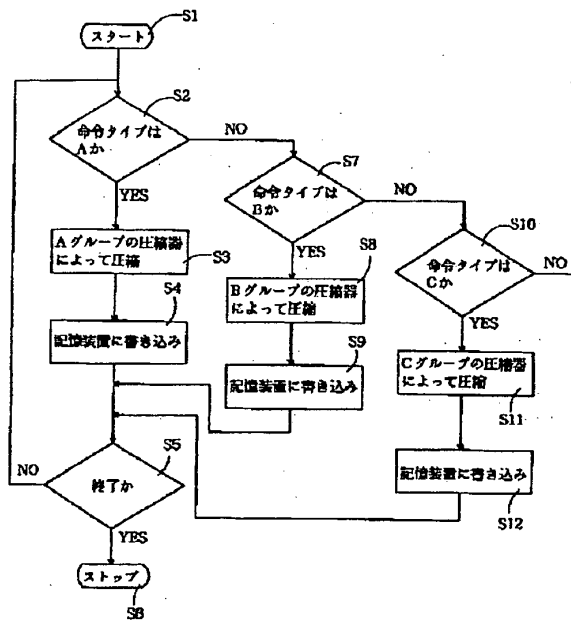


(C)

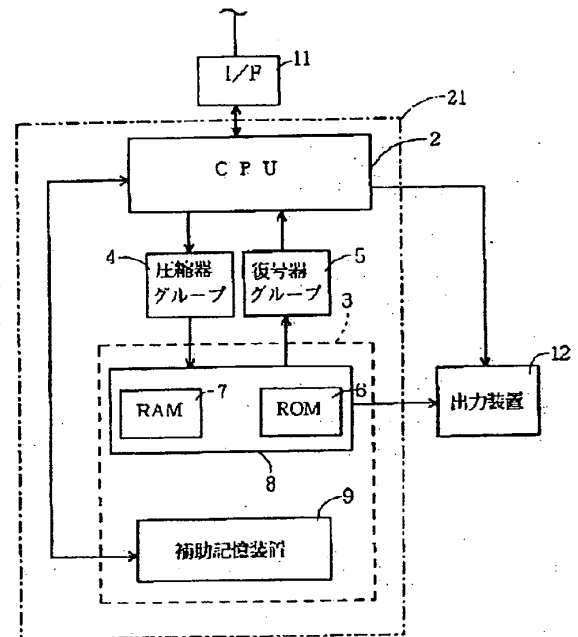
最新出現表

色順位	カラーシンボル
1位	C4
2位	C3
3位	C6
4位	C5
5位	C2
6位	C0
7位	C8
...	...
16位	Cn

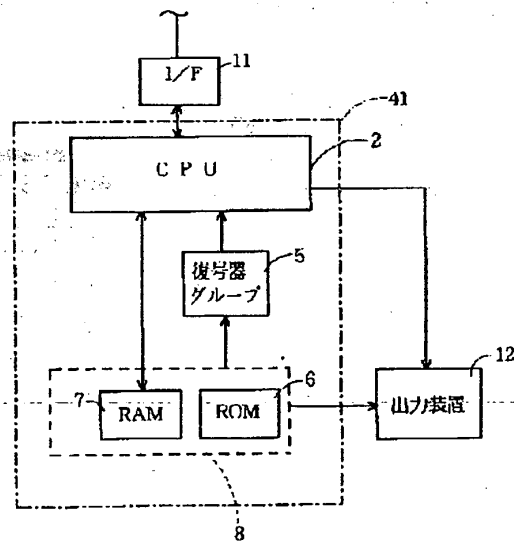
【図 4】



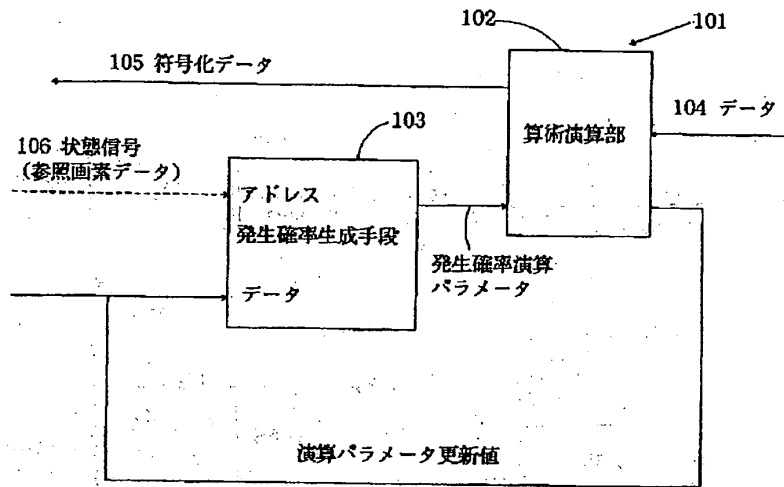
【図 7】



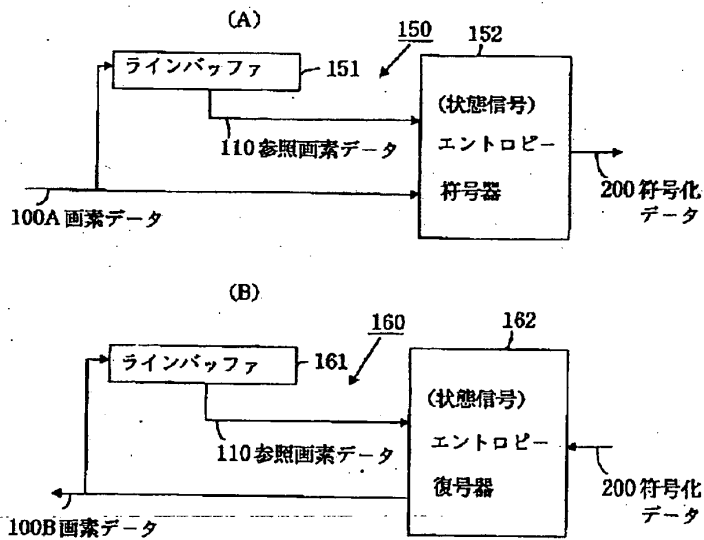
【図 8】



【図 9】



【図 10】



【図 1 2】

